04

Referencias y encapsulamento entre listas

Transcripción

[00:00] Hola. ¿Cómo están? Vamos a iniciar nuestra clase 7. En nuestra clase 7 vamos a ver referencias entre listas y cómo encapsular una lista también para que pueda ser inmutable. Por ejemplo, primero, para no confundir, en lugar de este modo de colocarle como nombre clase, vamos a cambiarle de nombre. Vamos a colocarle un sinónimo Aula. Vamos a hacer rename de todo lo que tenía clase list para aula.

[00:41] Refactoramos. Y una vez hecho esto, también no nos podemos olvidar de que nuestra clase curso aquí tenemos addClase. Entonces aquí sería addAula. Listo. Ejecutamos de nuevo nuestra clase 6 y perfecto, funcionó. Ahora vamos a duplicar nuestra clase 6 y vamos a colocar clase 7. Ahora en nuestra clase 7 vamos a hacer lo siguiente.

[01:22] Primero, aquí en nuestra clase curso, hemos creado un método para encapsular, que es el método addAula. Aquí estamos encapsulando el add de nuestra lista. Pero entonces vamos a hacer lo siguiente. Tenemos aquí nuestra lista de cursos y adicionamos curso1. Ahora imaginemos, tengo una lista de Aula, vamos a colocar aquí aulaList = curso1.

[02:04] Pero esto primero vamos a realizarlo en nuestro proyecto de la clase 7. Vamos a hacer aquí curso1.getAulaList(). ¿Esto qué me devuelve? Una lista de aulas. Entonces vamos a colocar aquí List vamos a colocarle aquí aulaList = . Importamos nuestra interface list y hacemos lo siguiente.

[02:41] Vamos a colocar aulaList.add y vamos a adicionar una nueva aula. Vamos a colocarle aquí new Aula. En este new Aula vamos a colocar aquí "Inmutable". Para esto, aquí hice lo siguiente. En nuestra clase curso, aquí coloque la clase Collections.unmodifiableList. Para esto, voy a remover y después voy a mostrarles lo siguiente.

[03:28] Ejecutamos nuestra clase 7. ¿Qué tenemos? ArrayList, List, LinkedList, Inmutable. Adicionó la lista. Entonces, si nosotros queremos que esta lista sea inmutable, ahí es donde vamos a utilizar nuestra clase Collections.unmodifiableList, para que no sea modificada esa lista.

[03:57] Entonces eso quiere decir que la lista que nosotros vamos a obtener haciendo get, no vamos a poder modificarla. Una vez hecho esto, tenemos error. ¿Por qué? Porque no podemos adicionarla. Para eso tenemos que utilizar, ¿qué cosa? Nuestra clase encapsulada. ¿Cuál es nuestra clase encapsulada? addAula.

[04:21] Entonces tenemos que utilizar el get, va a seguir siendo get. Aquí podemos imprimir nuestra lista de aulas por ejemplo, y ahí está. Pero ahora si queremos poner una más, tenemos que poner curso1.addAula, en new Aula vamos a colocarle "Inmutable". Ejecutamos y tenemos nuestra lista y tenemos modificado aquí en nuestra lista.

[05:07] Para esto es muy importante saber que el método getAulaList, como ya está para no poder modificarla, no vamos a poder hacer mas nada con esa lista. Solamente podemos utilizar el método add Aula, que eso es muy importante tenerlo presente. Entonces aquí imaginémonos queremos adicionado otra nueva aula, utilizamos nuestra clase curso1 y adicionamos otra nueva aula.

[05:36] Esa es la parte de encapsulamiento y la parte principal de tener una lista inmutable, una lista no modificable. Eso sería todo en nuestra clase 7, nos vemos en la siguiente clase. Muchas gracias.